package com.twitter.frigate.pushservice.util

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.frigate.common.base.CandidateDetails

import com.twitter.frigate.common.base.CandidateResult

import com.twitter.frigate.common.base.Invalid

import com.twitter.frigate.common.base.OK

import com.twitter.frigate.common.base.Result

import com.twitter.frigate.common.base.TweetAuthor

import com.twitter.frigate.common.base.TweetCandidate

import com.twitter.frigate.pushservice.model.PushTypes.PushCandidate

import com.twitter.frigate.pushservice.model.PushTypes.RawCandidate

import com.twitter.frigate.pushservice.params.PushFeatureSwitchParams.ListOfAdhocIdsForStatsTracking

class AdhocStatsUtil(stats: StatsReceiver) {

private def getAdhocIds(candidate: PushCandidate): Set[Long] =

candidate.target.params(ListOfAdhocIdsForStatsTracking)

private def isAdhocTweetCandidate(candidate: PushCandidate): Boolean = {

candidate match {

case tweetCandidate: RawCandidate with TweetCandidate with TweetAuthor =>

tweetCandidate.authorId.exists(id => getAdhocIds(candidate).contains(id))

case \_ => false

}

}

def getCandidateSourceStats(hydratedCandidates: Seq[CandidateDetails[PushCandidate]]): Unit = {

hydratedCandidates.foreach { hydratedCandidate =>

if (isAdhocTweetCandidate(hydratedCandidate.candidate)) {

stats.scope("candidate\_source").counter(hydratedCandidate.source).incr()

}

}

}

def getPreRankingFilterStats(

preRankingFilteredCandidates: Seq[CandidateResult[PushCandidate, Result]]

): Unit = {

preRankingFilteredCandidates.foreach { filteredCandidate =>

if (isAdhocTweetCandidate(filteredCandidate.candidate)) {

filteredCandidate.result match {

case Invalid(reason) =>

stats.scope("preranking\_filter").counter(reason.getOrElse("unknown\_reason")).incr()

case \_ =>

}

}

}

}

def getLightRankingStats(lightRankedCandidates: Seq[CandidateDetails[PushCandidate]]): Unit = {

lightRankedCandidates.foreach { lightRankedCandidate =>

if (isAdhocTweetCandidate(lightRankedCandidate.candidate)) {

stats.scope("light\_ranker").counter("passed\_light\_ranking").incr()

}

}

}

def getRankingStats(rankedCandidates: Seq[CandidateDetails[PushCandidate]]): Unit = {

rankedCandidates.zipWithIndex.foreach {

case (rankedCandidate, index) =>

val rankerStats = stats.scope("heavy\_ranker")

if (isAdhocTweetCandidate(rankedCandidate.candidate)) {

rankerStats.counter("ranked\_candidates").incr()

rankerStats.stat("rank").add(index.toFloat)

rankedCandidate.candidate.modelScores.map { modelScores =>

modelScores.foreach {

case (modelName, score) =>

// mutiply score by 1000 to not lose precision while converting to Float

val precisionScore = (score \* 100000).toFloat

rankerStats.stat(modelName).add(precisionScore)

}

}

}

}

}

def getReRankingStats(rankedCandidates: Seq[CandidateDetails[PushCandidate]]): Unit = {

rankedCandidates.zipWithIndex.foreach {

case (rankedCandidate, index) =>

val rankerStats = stats.scope("re\_ranking")

if (isAdhocTweetCandidate(rankedCandidate.candidate)) {

rankerStats.counter("re\_ranked\_candidates").incr()

rankerStats.stat("re\_rank").add(index.toFloat)

}

}

}

def getTakeCandidateResultStats(

allTakeCandidateResults: Seq[CandidateResult[PushCandidate, Result]]

): Unit = {

val takeStats = stats.scope("take\_step")

allTakeCandidateResults.foreach { candidateResult =>

if (isAdhocTweetCandidate(candidateResult.candidate)) {

candidateResult.result match {

case OK =>

takeStats.counter("sent").incr()

case Invalid(reason) =>

takeStats.counter(reason.getOrElse("unknown\_reason")).incr()

case \_ =>

takeStats.counter("unknown\_filter").incr()

}

}

}

}

}